Physics of Plasmas ARTICLE scitation.org/journal/php

# Machine learning control for disruption and tearing mode avoidance

Cite as: Phys. Plasmas **27**, 022501 (2020); doi: 10.1063/1.5125581
Submitted: 28 August 2019 · Accepted: 1 January 2020 ·
Published Online: 3 February 2020



Yichen Fu,[1] David Eldon,[2] Keith Erickson,[3] Kornee Kleijwegt,[4] Leonard Lupin-Jimenez,[1] Mark D. Boyer,[3] Nick Eidietis,[2] Nathaniel Barbour,[5] Olivier Izacard,[1] and Egemen Kolemen[3,6,a)]

## AFFILIATIONS

[1]Department of Astrophysical Science, Princeton University, Princeton, New Jersey 08544, USA
[2]General Atomics, PO Box 85608, San Diego, California 92186-5608, USA
[3]Princeton Plasma Physics Laboratory, Princeton, New Jersey, 08543-0451, USA
[4]Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands
[5]Department of Physics, Yale University, New Haven, Connecticut 06516, USA
[6]Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, New Jersey 08544, USA

**Note:** This paper is part of the Special Collection: Invited Papers from the 2nd International Conference on Data-Driven Plasma Science.
[a)]**Author to whom correspondence should be addressed:** ekolemen@pppl.gov

## ABSTRACT

Real-time feedback control based on machine learning algorithms (MLA) was successfully developed and tested on DIII-D plasmas to avoid tearing modes and disruptions while maximizing the plasma performance, which is measured by normalized plasma beta. The control uses MLAs that were trained with ensemble learning methods using only the data available to the real-time Plasma Control System (PCS) from several thousand DIII-D discharges. A "tearability" metric that quantifies the likelihood of the onset of 2/1 tearing modes in a given time window, and a "disruptivity" metric that quantifies the likelihood of the onset of plasma disruptions were first tested off-line and then implemented on the PCS. A real-time control system based on these MLAs was successfully tested on DIII-D discharges, using feedback algorithms to maximize $\beta_N$ while avoiding tearing modes and to dynamically adjust ramp down to avoid high-current disruptions in ramp down.





## I. INTRODUCTION

Disruptions in tokamaks are a rapid loss of plasma confinement and plasma current. During a disruption, the plasma releases its thermal energy, and the plasma current rapidly decreases to zero. Rapid release of this energy during a disruption can potentially cause surface melting of plasma-facing components and high electromagnetic loads close to the design limits.[1] To prevent unnecessary maintenance, the vast majority of potential disruptions must be avoided or mitigated in large-scale tokamaks such as ITER. For disruption avoidance, a diagnostic system that provides adequate information to guide the steering of tokamaks is required.[2] The complexity of disruptions stands as a barrier to developing a comprehensive physical model.

To detect and indicate an impending disruption, the majority of large tokamaks use a physics-based mixture of signals:[3] in JET one of the detection thresholds is set on the n = 1 locked mode amplitude; on ASDEX Upgrade, detection of an impending radiative (detached) disruption is monitored through the divertor temperature and radiation;[4] on TEXTOR, a heterodyne electron cyclotron emission diagnostic was developed to detect an $m/n = 2/1$ disruption precursor.[5] However, due to the complexity of disruptions, more sophisticated disruption detection and prediction methods have been developed. One category of these predictors is developed based on physics models. For example, the Disruption Event Characterization and Forecasting Code (DECAF)[6] analyzes tokamak data to determine chains of events that lead to disruptions and to forecast their evolution, providing a quantitative and deterministic predictor for disruptions. Another predictor[7] has been built up based on diagnostic data of the high-$\beta$ spherical torus NSTX. The disruptive threshold values of many signals are examined, and a novel means of combining multiple threshold tests has been developed.

Another broad category of predictors has been developed using machine learning. Machine learning algorithms (MLAs) are valuable tools for both classification and regression tasks. Researchers provide the algorithm with a training set of input signals and target values, and the algorithm determines patterns that best map the input signals to the target values for each case in the training set. Thus, after

appropriate training, the algorithm will be able to recognize precursors of instabilities from complex signals and detect approaching disruptions. A large amount of disruption prediction studies employ artificial neural networks (ANN), which mimic the brain's network of neurons by sending input signal data through multiple layers of artificial neurons, functions that convert input signals into an activation signal. Over the past two decades, researchers at TEXT,[8] DIII-D,[9] ASDEX,[10] JET,[11,12] JT-60U,[13] and ADITYA[14] have conducted proof of concept studies employing ANNs to detect disruptions by using diagnostic data available in real-time as input signals. Recently, a disruption predictor combining recurrent and convolutional neural networks has been developed by Julian *et al.*[15] The neural network has also been applied to analyze the tearing mode (TM) on JET.[16] Other machine learning methods, such as support vector machines[17] and discriminant analysis,[18] have also been applied. The support vector machine[19] and anomaly detection method[20] have been implemented in real-time in JET. A different approach called classification and regression trees (CART),[21] which could estimate the relative importance of the input features and is more useful for data interpretation,[22] was tested on JET.[23] Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one.[24] A large category of techniques, usually called ensemble learners, can combine the strength of more than one tree to give a better prediction.[25] A disruption predictor using random forests, one of the ensemble methods, has recently been developed on DIII-D and Alcator C-Mod by Rea *et al.*[26]

In this paper, we present a disruption prediction and avoidance algorithm for DIII-D based on ensemble methods. In contrast to Ref. 26 using 10 signals at different time moments, we made use of a broader range of signals and calculated the mean, trend, and variance within a specific time window. In addition, we focus more on using the machine learning method to develop real-time control algorithms in two different scenarios: tearing mode avoidance and ramp down control. In Sec. II, we briefly describe the algorithm used in this paper and the method to evaluate the algorithms. The construction of our database for the two scenarios, as well as the input signals and output target values, will be presented in Sec. III. The results of our predictors are discussed in Sec. IV. In the first scenario, an algorithm was trained to predict the likelihood of a $m/n = 2/1$ tearing mode, which is believed to be a precursor to major disruptions.[27] By monitoring the predicted likelihood and modulating neutral beam injection, the plasma can be kept within the stable region. In the second scenario, an algorithm was trained to predict incoming disruptions instead of its precursor, which means when detected, the plasma is already in the unstable region. This prediction was applied in the ramp-down control to change the ramp-down speed in order to prevent disruption. The detailed description of two scenarios and preliminary experimental results will be presented in Secs. V and VI, respectively.

## II. MACHINE LEARNING MODEL

In this section, we introduce the machine learning algorithm used in our experiment. The models are implemented in the framework of OMFIT[28] and loaded from python library scikit-learn.[29] In order to run in real-time on DIII-D, the algorithm was first transferred into MATLAB and then converted into a C library function using the Embedded MATLAB Coder to run in the plasma control system (PCS).[30]

### A. Decision trees

Decision trees are a non-parametric method, which does not make assumptions about the form of the function to be learned. It predicts the target value by learning some decision rules from the training data. Decision trees are of two main types, classification trees and regression trees. Classification trees provide a finite number of outputs; regression trees give piece-wise constant output. A simple regression tree is shown in Fig. 1. A more complicated tree would produce a nearly continuous output. In this paper, we use regression trees to predict a target value, disruptivity or tearability, which is roughly the likelihood of disruption or tearing mode occurring, respectively. These concepts will be explained with more details in Sec. III C.

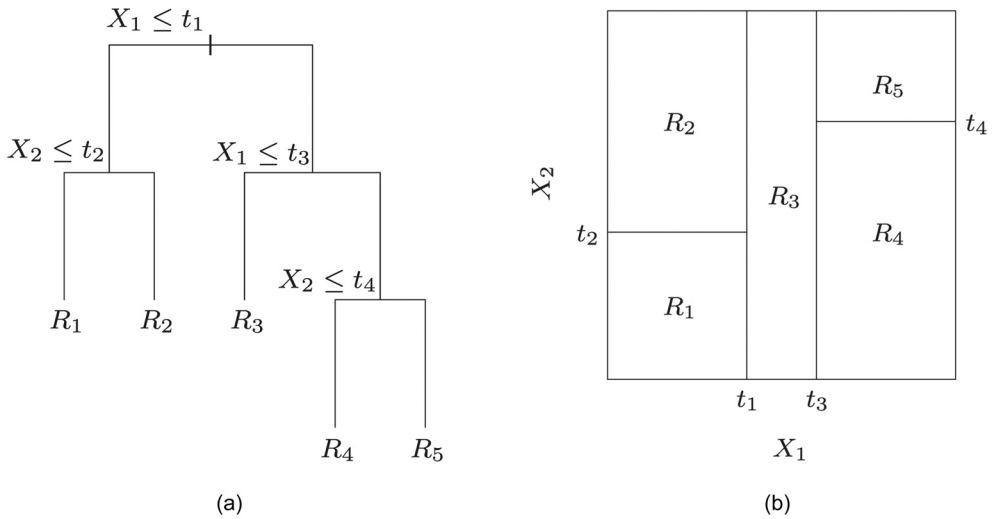


**FIG. 1.** Demonstration of a decision tree:[31] (a) shows a simple decision procedure from top to bottom for two features $X_1, X_2$; (b) shows how a function of two variables is defined based on the decision tree.

Regression tree algorithms use recursive partitioning to group similar cases by minimizing the total variance across all dimensions of the input data. A basic regression tree is grown as follows. All of the input cases enter the tree at the base. The algorithm then successively performs binary splits, separating the cases that have a value higher than the chosen split value from the cases with a value less than the split value. In order to determine the split value, the algorithm scans across features to choose the feature and value, which could reduce the total variance across the entire set. This binary splitting process is repeated for each subset of cases that are created successively until each split contains a minimum of two cases.

One of the advantages of decision tree methods is that they can give the relative importance of all input variables.[32] Every time the split is done according to some variables, the variance of the two descendant nodes is less than the parent node. Adding up all the variance decrease for that particular variable over the tree, we can obtain the so-called Gini importance for this variable. After computing the importance of every variable, we can compute their relative importance. The calculation of Gini importance is done by the standard function feature_importances_ in scikit-learn. The importance given by our algorithms could potentially provide some insight into the physical reason for the disruptions.

## B. Ensemble methods

To have better prediction capability, we applied ensemble methods to strengthen our algorithm. The idea of ensemble methods is to build a prediction model by combining the strengths of a collection of simpler base estimators.[24] Here, we compared four different ensemble methods: Bagging, random forests, extremely randomized trees, and AdaBoost. The first three belong to averaging methods and the last one is a boosting method. The averaging method creates several independent estimators and then averages their predictions to decrease variance. In contrast, the boosting method builds estimators sequentially and tries to reduce bias.

Bagging applies bootstrap sampling[33] to create a large number of different copies of the original training set. For each copy, the bootstrap procedure randomly picks elements to construct a new training set with the same number of elements as the original set. However, some elements (around $1/e = 37\%$) in the original set do not appear in the new set and some elements repeat several times. Then, the algorithm obtains different versions of a base predictor using these different copies as new training sets. When predicting a testing set, it aggregates averages over the different versions and gives its mean value.[34]

Random forests are a substantial modification of bagging that builds an extensive collection of de-correlated trees, and then averages them.[24,35] Usually, random forests use bootstrap methods, like the bagging method, to create different new training sets. However, when splitting a node, it chooses the best split among a random subset of the features instead of all. By using this method, different versions of predictors are generated and the final output is also the average of all those predictors.

Extremely randomized trees build an ensemble of decision trees and add randomness during the splitting.[36] When splitting a node, it will first randomly choose a subset of features. Unlike random forest looking for the best split among these subsets, extremely randomized trees randomly generate a set of different splits. Then, the split of this node is chosen to be the best split among all those random ones. A large number of such random decision trees are created and again, the final output is the average of all those trees.

AdaBoost is a method to improve the accuracy of any base algorithm.[37] AdaBoost uses the training set to fit the base estimator iteratively, maintaining a set of weights over the training set and modifying this set of weights during each iteration. Initially, those weights are all equal. At each iteration, the weight of those data incorrectly predicted by the previous estimator is increased, whereas the weight of those correctly predicted data is decreased. After $N$ iterations, $N$ different estimators are created, and the output of AdaBoost is the weighted mean according to the final weights of the training data.

## C. Evaluation of algorithms

For a classification problem with two classes, each instance can be labeled as positive and negative, or in our case, disruptive or non-disruptive. If an instance is positive and it is predicted as positive, then this prediction is counted as true positive (TP); if it is predicted as negative, then it is counted as false negative (FN). If an instance is negative and it is predicted as negative, then it is counted as true negative (TN); if it is predicted as positive, it is counted as false positive (FP). The true positive rate (TPR) is the number of true positive predictions out of the total number of positive instances; the false positive rate (FPR) is the number of false positive predictions out of the total number of negative instances. A two-by-two confusion matrix, shown in Table I, can be constructed to represent all four cases.

In our algorithm, the classification of positive (disruptive) or negative (non-disruptive) is based on a threshold, which will be discussed in detail in Sec. III C. In order to evaluate the performance of these kinds of algorithms, one common method is to use a receiver operating characteristics (ROC) graph.[38] ROC graphs are a two-dimensional graph where the TPR is the Y-axis and the FPR is the X-axis, for example, Fig. 5(d). For a classifier parametrized by a threshold, we can imagine varying the threshold and tracing out a curve in the ROC graph. Each point on this curve represents the TPR and FPR of the algorithm for that value of the threshold. The whole curve describes

**TABLE I.** Confusion matrix.

| | | True class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **T**rue **p**ositive | **F**alse **p**ositive | TPR = TP/(TP + FN) |
| | Negative | **F**alse **n**egative | **T**rue **n**egative | FPR = FP/(FP + TN) |
| | Total | TP + FN | FP + TN | |

the trade-offs between benefits (TPR) and costs (FPR) of the algorithm. The more "northwest" the curve, the better the algorithm.

## III. DATA REDUCTION

### A. Sequence of disruption and database selection

A large fraction of disruptions start from some form of plasma instability. An accepted sequence of events in a disruption[3] is plasma energy lost (Thermal Quench); plasma moves and hits the wall; impurities enter; plasma cools and becomes highly resistive; plasma current is lost (Current Quench).

A schematic diagram of the plasma state in some parameter space is shown in Fig. 2. In order to have stable fusion, the plasma has to be in stable equilibrium. However, in reality, to have a high-performance plasma, it may be at risk of crossing the stability limit, which would end a plasma discharge and cause damage to the device. Thus, in practice, it is essential to find out what is the current state of the plasma and predict what the plasma state would be in the future. If the plasma is currently stable but about to cross the stability limit, we can manually change some plasma parameters and avoid the incoming instability; if there is already an instability growing and the plasma is about to disrupt, we need to ramp down the plasma immediately or trigger some mitigation systems.

For these two different situations, we need two different algorithms to detect and predict them. To prevent the plasma from crossing the stability limit, we chose the most important instability, tearing mode,[39] as our main feature to be detected. We created a tearing mode database using 1970 shots from DIII-D 2014–2017 campaigns, which contain 2/1 tearing modes (TM). The shots were selected with strong $n = 1$ mode satisfying: 1. $\text{n1rms}^2 > 10$ G, 2. mode duration $> 50$ ms. The moment of a tearing mode happening is represented by the first-time $n = 1$ mode satisfying the constraints above. We also randomly collected 2000 non-disruptive shots during the same campaigns whose maximum $n = 1$ mode amplitude $< 5$ G into our tearing mode database as our non-tearing mode shots. Then, to detect plasma disruptions, we created a disruption database from DIII-D 2013–2017 campaigns. All disruptive shots during that period with at least 1-s flat-top were collected, added up to 1906 disruptive shots. Another 2000 non-disruptive shots from the same campaigns was randomly collected into our disruption database. The exact moment of disruption is estimated by the steepest change of plasma current over time, i.e., $\max_t |\frac{\partial I_p}{\partial t}|$. For both databases, we want our algorithm to predict a particular event, the occurrence of tearing mode or disruption, before it happens. For convenience, we collectively refer the moment these events occur as "target moment." When training for disruption prediction, the algorithm would be trained with disruptive and non-disruptive shots; when training for tearing mode prediction, the algorithm would be trained with tearing mode shots and non-tearing mode shots.

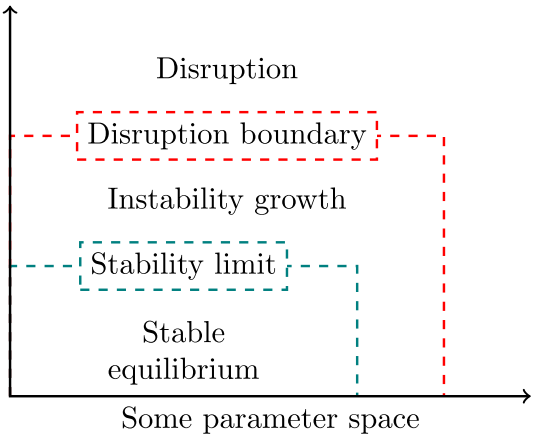


**FIG. 2.** Plasma state in some parameter space. The blue dashed line is the stability limit, inside which the plasma is in a stable equilibrium. The red dashed line is the disruption boundary, outside which the plasma will disrupt. Between these two lines, instabilities grow in the plasma but do not cause disruption yet.

### B. Parameters and data preprocessing

For each shot in our database, we download the following signals, shown in Table II, from MDSplus[40] and rtEFIT[41] in DIII-D, which are all real-time available:

For disruptive shots in the disruption database and all shots in the tearing mode database, the data from the beginning of the flat-top to 10 ms before the target moment ($t_0$) were used. The data for non-disruptive shots were chosen from 2 s before ramp-down and included the ramp-dawn phase. In this data set, we divide the data for each shot into 200 overlapping frames; each of them is 100 ms long. The distance between the center of two adjoint frames is 10 ms. The predictions are made by using the information obtained from a 100 ms interval. Each frame is divided again into 6 sub-frames: the whole frame, two half frames, and three third frames. In each sub-frame, the mean, variance, and trend of each signal are calculated. Here, the trend is given by the slope of the linear fit of each signal. If the calculation failed due to missing signals, the value would be set to 0. In total, $18 (= 3 \times 6)$ values are extracted as the features of each signal in each frame, as the input to our prediction algorithms.

As pointed out in Ref. 26, some differences between disruptive shots and non-disruptive shots can be directly identified by the distribution of some parameters. Here, the distribution of $li$ and $1/q_{95}$ at $\Delta t = t - t_0 = -250$ ms for both disruptive and non-disruptive shots is shown in Fig. 3. Compared to non-disruptive shots, disruptive shots have a significantly larger fraction of distribution at higher $li$ and lower $q_{95}$. Each of these signals is not able to determine whether a shot is disruptive or not. However, the increased prevalence of disruption at particular values implies that by combining the value of multiple signals, machine learning algorithms have the potential to predict whether a shot is disruptive or not.

### C. Target value

Since the typical energy confinement times at DIII-D are on the order of 100–300 ms,[42] we attempt to predict our target moments 250 ms before they happen. The comparison of different prediction windows is shown later in Sec. IV B. In order to train a regression model, a target value needs to be provided to the training set that the algorithm can refer to. Similar to the approach in Ref. 43, our target value is defined by a sigmoid function:

$$\text{Target value} = \frac{1}{1 + e^{(\Delta t - x)/t_w}}.$$

Here, $t_0$ is when target moment happens, $t$ is the time at the end of the frame, and $\Delta t = t - t_0$ is the remaining time at the end of each frame

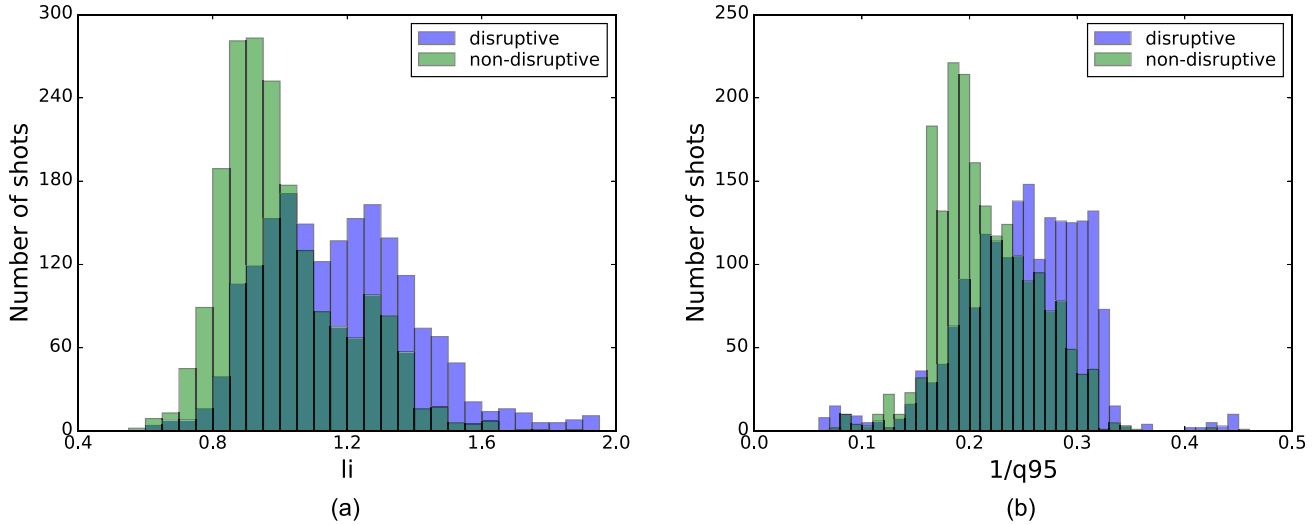


**FIG. 3.** Distribution of $li$ and $1/q_{95}$ for both disruptive shots and non-disruptive shots. For disruptive shots, data were collected 250 ms before disruption; for non-disruptive shots, data were collected 250 ms before ramp-down.

up to the target moment; $x = 250$ (ms) is the prediction window, which defines how much time before disruption we want our algorithm to make a prediction; $t_w = 10$ (ms) is the width of the sigmoid function, which is equal to the distance between two adjoint frames. The shape of the sigmoid function is shown in Fig. 4.

When the time is far from the target moment, the target value is almost 0; as the time approaches the target moment, the target value gradually increases and goes to 1. For the tearing mode database, we call the target value tearability; for the disruption database, we call the target value disruptivity. To calculate the TPR and FPR from our prediction, we denote frames with target value $> 0.5$ as positive, i.e., a prediction that the end time of the frame is within the prediction window, and the rest of the frames are labeled as negative. After being trained, the output of our algorithm will be a real value from $0 \sim 1$, which can be roughly regarded as the "probability" that the target moment will occur in the next 250 ms. This probability is then compared to a threshold value, and if it exceeds this threshold, we predict the target moment will occur, vice versa.

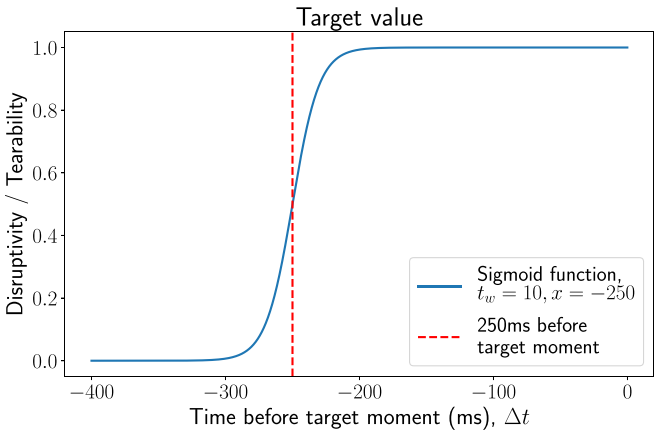


**FIG. 4.** The sigmoid function used for target value creation.

## IV. ANALYSIS OF DISRUPTION PREDICTION RESULTS

### A. Overview of the analysis procedure

A schematic overview of the analysis procedure is demonstrated in Fig. 5. Figure 5(a) represents a typical structure of one regression tree. First, the forest of decision trees is used to make a disruptivity prediction. Arrow "A" in Fig. 5(a) represents this process. Since regression algorithms are used in this paper, the output of these decision tree forests will be a scalar disruptivity value. Figure 5(b) is an example of disruptivity as a function of time. The disruptivity is given by our machine learning algorithms and averaged over the test disruptive set and non-disruptive set. Using a threshold on the scalar value, a differentiation between disruptive and non-disruptive discharges can be made. The threshold is indicated by the dashed red line labeled "B." Arrow "C" represents the relation of the threshold in two different graphs. Figure 5(c) displays the ratio of true positives and false positives, calculated in the testing set, as a function of threshold. Plotting the true positives as a function of false positives as represented by arrows "D" results in Fig. 5(d) the true positives vs false positives, giving the ROC curve described in Sec. II C.

### B. Prediction results of the database

In a real-time experiment, we need to use an algorithm trained by all the databases. However, to test our algorithm and choose a reasonable threshold, we need to split our data into training and testing sets. Thus, we randomly chose 80% of the shots to be the training set and put the remaining 20% shots, which have never been used during the training, into the testing set. Meanwhile, we keep roughly 50% of disruptive shots in both the training set and testing set. Here, we show the prediction results in Fig. 6 using the ROC curve for the disruption database. In this ROC curve, the true positive rate and false positive rate are counted by the number of frames. From the ROC curve, we

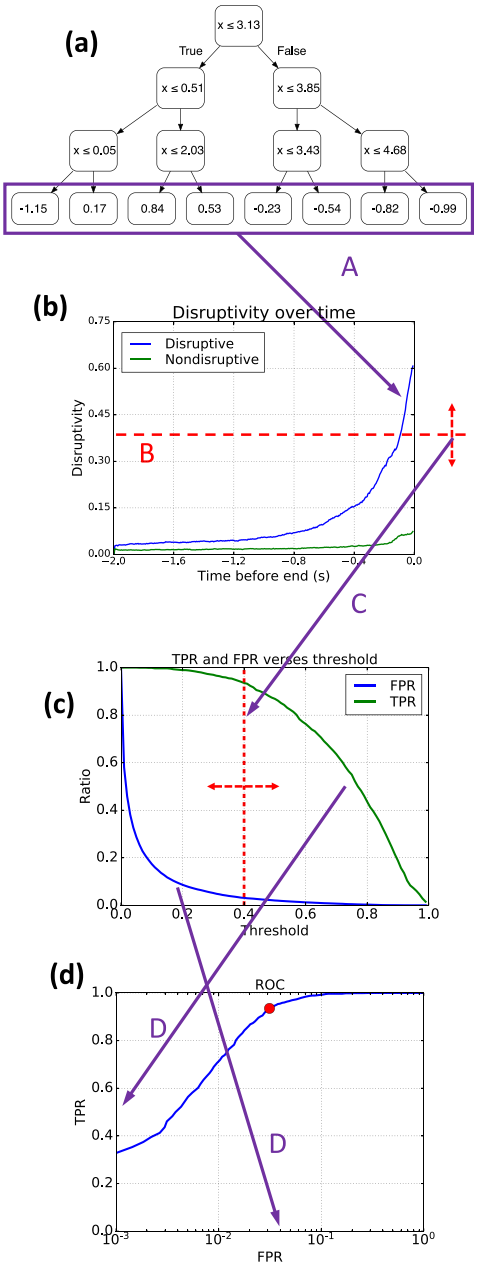


**FIG. 5.** (a) Top of example of one decision tree with depth of two, "left side arrow" is the true decision and "right side arrow" is the false decision. (b) Disruptivity prediction over time plot on both disruptive (blue) and non-disruptive (green) shots. (c) True positive rate (TPR, green line), false positive rate (FPR, blue line) as a function of threshold. (d) ROC graph, true positives rate [y-axis] as a function of false positives rate [x-axis]. Arrow "A" represents the use of the decision tree forest to create a disruptivity prediction over time for both disruptive and non-disruptive shots. Using a threshold (dashed red line "B"), a classification of the shot can be made. The true positives and false positives for a range of thresholds between 0 and 1 are displayed in (c). By arrow "B," the relation and direction between (b) and (c) of changing the threshold is given. Arrows "D," using the two lines in (c) a ROC graph, the true positives as a function of the false positives, as in (d) is formed for more insight in the relation between the two.

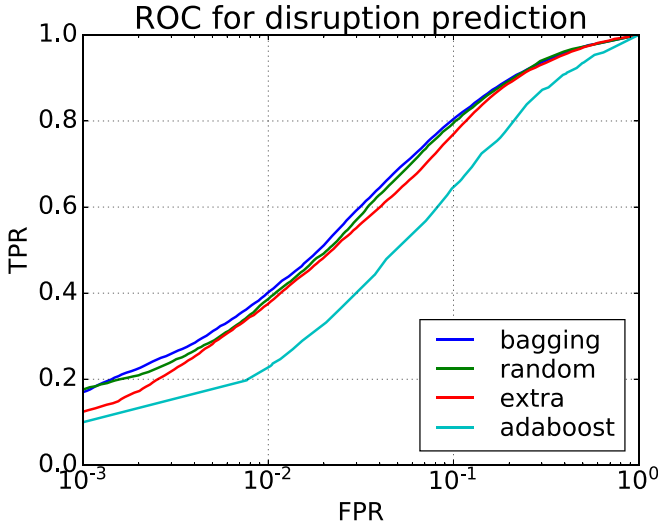


**FIG. 6.** ROC curve for disruption prediction. The dark blue line is extreme randomized tree, the green line is Adaboost, the red line is bagging, and the light blue line is random forest.

can see that for disruption prediction, bagging has the best prediction capability, having <10% false positive rate while >80% true positive rate. The Adaboost method is the worse among all four methods for disruption prediction.

In order to examine the relation between the prediction capability of our algorithm and the prediction window, different prediction windows were used for bagging method training with the disruption database. The training results are shown by the ROC curve in Fig. 7. We can see that with the same FPR, the TPR increases almost monotonically when the prediction window decreases. When prediction window = 50 ms, the bagging method has >95% TPR when FPR ∼ 10%. This

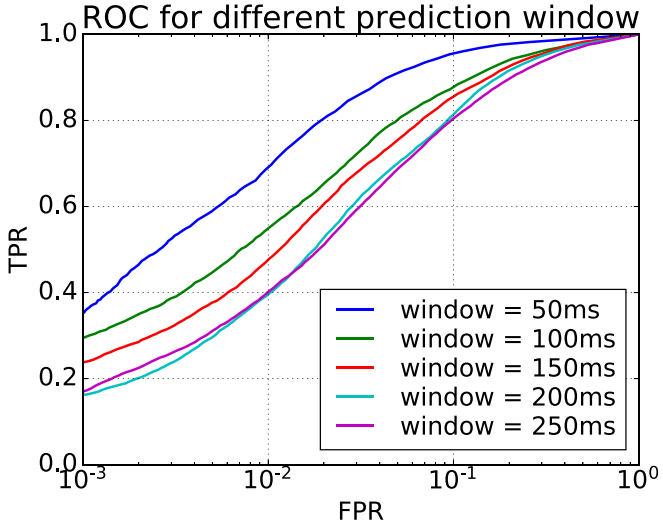


**FIG. 7.** ROC curve for different prediction windows using bagging method. Different colors represent different prediction windows from 50 ms to 250 ms.

trend is anticipated since closer to disruption, the state of plasma would be closer to the disruption boundary. Nevertheless, to predict disruptions early enough to take action and avoid them, we chose a prediction window = 250 ms.

The average of disruption prediction over time is shown in Fig. 8. In this graph, the average prediction of bagging on disruptive shots and non-disruptive shots is shown. From Fig. 8, we can see that when far away from disruption, the prediction on disruptive shots and non-disruptive shots is similar. However, when disruption is close, the prediction becomes different. When the time goes over the prediction window (250 ms before disruption), the predicted disruptivity for disruptive shots grows significantly, while the predicted disruptivity of non-disruptive shots remains relatively low. Thus, it is easy and feasible to use a threshold (for example, 0.25) to distinguish disruptive and non-disruptive shots.

As mentioned in Sec. II A, the algorithm can provide the relative importance for each feature. To calculate the total importance of a signal, we added up all contributions of 18 features extracted from each signal. The four most essential signals given by our disruption predictors are shown in Table III.

### C. Prediction results on the whole shot

In order to measure the prediction of our algorithm on a whole shot, we still test our algorithm on the testing data described in Sec. IV B. However, instead of defining TPR and FPR based on the number of frames, we need to define it based on the number of shots. After choosing a certain threshold, if the disruptivity exceeds this threshold for five consecutive frames, we state that our algorithm predicting this shot will disrupt and this moment is defined as the alarm time, $t_A$. Then, the TPR is calculated by the number of disruptive shots that are predicted to disrupt over the total number of disruptive shots. The FPR is calculated by the number of non-disruptive shots that are predicted to disrupt over the total number of non-disruptive shots. The median alarm times are calculated over all true positive predictions at each threshold. The prediction results using the bagging method are shown in Fig. 9. As shown in this figure, when the threshold is too low, the algorithm captures all disruptive shots but misclassifies too many non-disruptive shots and predicts disruption too far away from when it actually happens. If the threshold is too high, then the algorithm could not identify some of the disruptive shots and would give too short alarm time. In order to strike a balance, we use the median value instead of the mean value to indicate the expected alarm time. We choose the threshold to be 0.25, where our algorithm has around 95% TPR and 15% FPR with the median alarm time around 320 ms. It is reasonable since we chose the prediction window to be 250 ms.

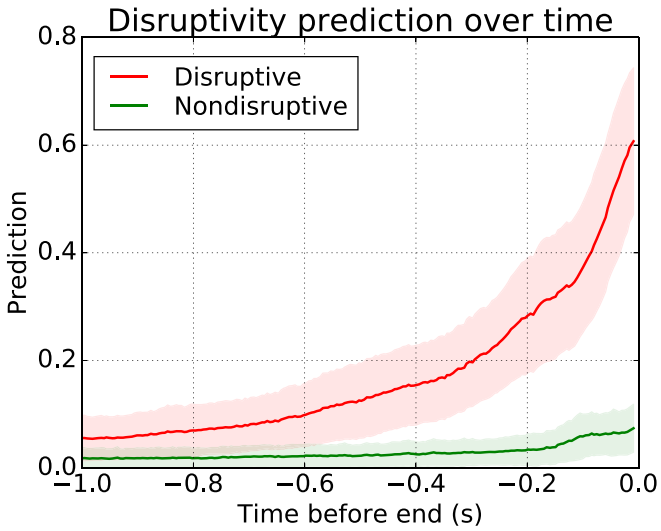


**FIG. 8.** The average of disruption prediction over time by bagging. The x-axis is the time before the shot end. For disruptive shots, it is the time before disruption; for non-disruptive shots, it is the time before ramp-down. The y-axis is the prediction of disruptivity. The shaded region indicates the standard deviation over all testing shots.

**TABLE II.** List of signal used in our database.

| Symbol | Name |
|---|---|
| $I_p$ | Plasma current |
| $I_p - I_{p,target}$ | Plasma current minus target current |
| $li$ | Normalized internal inductance |
| $\beta_N, \beta_p, \beta_t$ | Normalized beta, poloidal, toroidal beta |
| $W_{\mathrm{MHD}}$ | Plasma energy |
| $q_0, q_{\min}, q_{95}$ | Safety factor at different positions |
| $V_{\mathrm{loop}}$ | Loop voltage |
| $V$ | Plasma volume |
| $a_{\mathrm{minor}}$ | Minor radius |
| $R, Z$ | R, Z position of magnetic axis |
| $\langle n_e \rangle$ | Line averaged electron density |
| $V_{\mathrm{rot,edge}}, V_{\mathrm{rot,core}}$ | Plasma edge and core rotation |
| $\kappa, \delta_t, \delta_b$ | Plasma kappa, top, bottom triangularity |
| $P_{\mathrm{rad}}$ | Radiation power |
| $d_{\mathrm{sep}}$ | Distance of the separatrices |
| $\chi^2$ | Magnetic chi-square |
| $B_{\mathrm{rad}}$ | Radial magnetic field |
| n1rms, n2rms | Root-mean-square amplitude of magnetic fluctuations corresponding to toroidal mode number n = 1,2 |

The distribution of alarm times at threshold = 0.25 is shown in Fig. 10. This gives a Levy-like one-sided disruption which can be described by median and variation around it. At the chosen threshold, the peak of distribution is located around $t_A = 300$ ms, which is close to our prediction window. A small portion of shots is predicted more than 1500 ms before disruption, which is effectively ignored in our

**TABLE III.** Four most important signals in disruption prediction and its relative importances.

| Signal | Importance |
|---|---|
| $li$ | 0.120 |
| $\beta_p$ | 0.111 |
| $I_p - I_{p,target}$ | 0.085 |
| $\langle n_e \rangle$ | 0.053 |

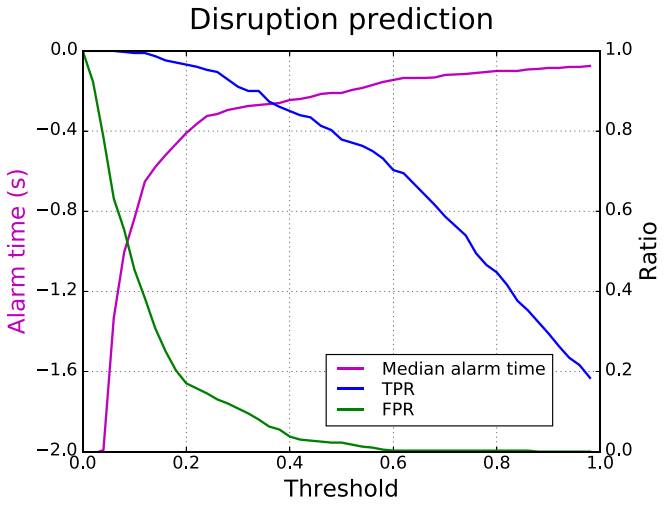


**FIG. 9.** TPR, FPR, and the median alarm times vs threshold for disruption prediction.

analysis by using the median alarm time as the indicator. For a normal distribution, the random variable within one standard deviation around its mean value counts roughly 68%. To represent the same idea, in the distribution of alarm times, ±34% around the median alarm time is shown in orange bins; the rest are shown in blue bins.

## V. DISRUPTION PREDICTION AND AVOIDANCE DURING THE RAMP DOWN PHASE

### A. Real-time disruption prediction

In Sec. IV, we have presented the results of predicting disruption using decision trees and ensemble methods. In order to demonstrate the feasibility of the algorithm in real-time, the algorithm trained with the disruption database has been implemented in PCS[30] in DIII-D and tested in real-time.

Two examples of detecting disruption in the ramp down phase are shown in Fig. 11. For each shot, the disruption happened during the ramp-down scenario. The first shot (174720) disrupted at $t = 2.58$ s at current of 0.39 MA and the second shot (174724) disrupted at $t = 3.01$ s at a current of 0.74 MA. ITER Baseline Scenario plasmas are most often simulated on DIII-D with plasma currents around 1.4 MA,[44] with a normalized current value of 1.42 for the ITER 15 MA. Since terminations on ITER below 3 MA are considered tolerable,[45] which maps to roughly $I_p = 0.3$ MA in DIII-D, both shots shown in Fig. 11 were above the tolerance threshold. For each shot, there were some instabilities, indicated by n = 1 RMS fluctuation amplitudes (n1rms) in Fig. 11(c), which grew at the shot that began at $t = 2.14$ s and $t = 2.41$ s. After these instabilities grew large enough, the mode lock happened at $t = 2.32$ s and $t = 2.53$ s, indicated by the radial magnetic field in Fig. 11(d), which eventually caused the shot to disrupt. During the disruption, the plasma current suddenly drops to zero along with a strong radiative power loss shown in Fig. 11(e). Note that in shot 174724, the disruptivity grew at the same time as the lock mode signal. In contrast, in shot 174720, the disruptivity did not grow as the lock mode signal. These two shots show that our algorithm was making the decision based on the combination of multiple signals instead of just looking at individual signals.

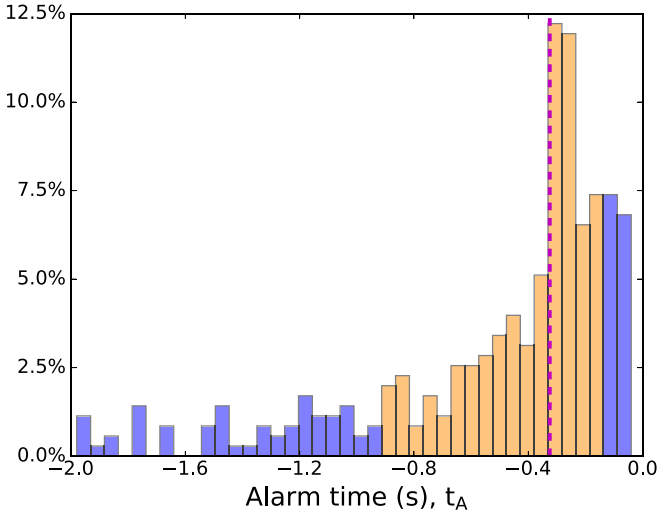


**FIG. 10.** The distribution of alarm times in disruption prediction at threshold = 0.25. The vertical dotted line represents the median alarm time 320 ms. Distributions within ±34% around the median value is shown by orange bins; the rest are shown by blue bins.

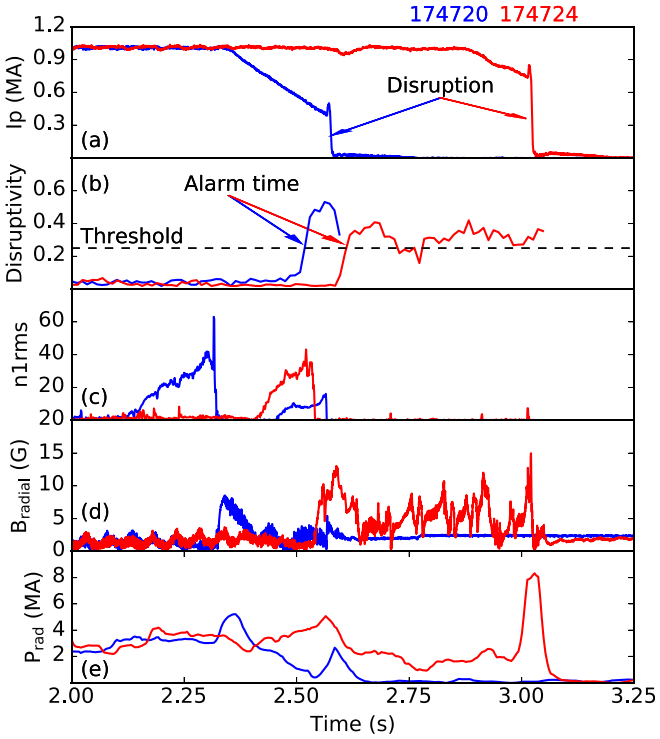


**FIG. 11.** Prediction of disruption at ramp down. The results of two shots are displayed in blue (174720) and red (174724), respectively. The signals shown are plasma current(a), n = 1 RMS fluctuation amplitudes(c), radial magnetic field (d) and radiated power(e). The machine learning prediction, disruptivity, is shown in (b) and the dashed line is the threshold to trigger the alarm.

During the whole shot, our machine learning algorithm was reading the information from diagnostics and predicting the disruptivity continuously. From Fig. 11(b), we can see that when far away from disruption, the disruptivity is relatively low ($< 0.1$). When closer to disruption, the disruptivity grew and exceeded the threshold (0.25) at $t = 2.51$ s and $t = 2.61$ s, respectively, giving an alarm time of 0.07 s and 0.4 s.

### B. Ramp down control

Disruptions in ramp down account for 41% of DIII-D disruptions since 2012. To prevent disruption, we designed a control algorithm to monitor the disruptivity during ramp down. The basic control logic is shown in Fig. 12. During the plasma ramp-down scenario, our algorithm read all the signals needed from real-time diagnostics and rtEFIT. Using the trained regression trees, the algorithm predicted the disruptivity. If the disruptivity increased above a certain threshold, we believe that some off-normal states would occur, and the plasma would disrupt after 250 ms. By using the DIII-D Off Normal Fault Response (ONFR)[46] capabilities, this event would trigger a fast ramp-down and terminate the plasma within 200 ms, no matter what the present plasma current was. By this strategy, we might be able to ramp down the plasma before the disruption happens; or even if the plasma does disrupt, we might be able to disrupt at a current lower than the tolerance threshold 0.3 MA, as discussed in Sec. V A.

For a real-time experiment, we need the algorithm to predict disruption at 250 ms before it happens. Thus, we chose the threshold to be 0.4, where the median alarm time is 250 ms, as shown in Fig. 9. The algorithm was tested from shots 176318 to 176349. The algorithm triggered 200-ms-ramp-down in 7 shots among all test shots, all of which eventually lead to no disruption or disruption at $I_p < 0.3$ MA. Among the shots which our algorithm did not trigger, only one of them (shot 176319) disrupted during ramp-down at $I_p = 0.36$ MA. Other shots either did not disrupt, disrupted at low $I_p$, or disrupted before ramp-down began. An example experiment (shot 176339) is shown in Fig. 13. In this shot, ramp down began at $t = 5.00$ s, and our ramp-down control began. Soon after ramp-down began, the plasma was changed from the divertor shape into limiter shape, as shown in Fig. 13(f), and maintained this status till the shot end. At first, the neutral beam power was kept the same as the flat-top and decreased at $t = 5.29$ s. As the shape changed, instabilities grew, indicated by $n = 1$ RMS fluctuation amplitudes (n1rms) in Fig. 13(c), and our algorithm predicted that the disruptivity was close to but not exceeding our

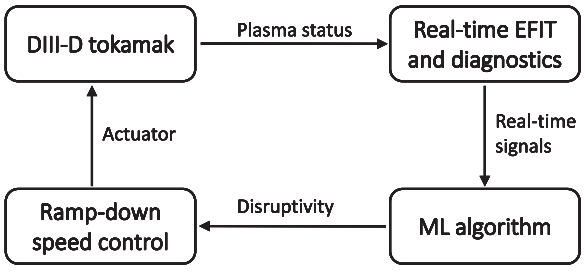


**FIG. 12.** The workflow of ramp-down control algorithm. Note that when machine learning algorithm predict disruption, only ramp-down speed, i.e., the speed of current decrease, would be changed.

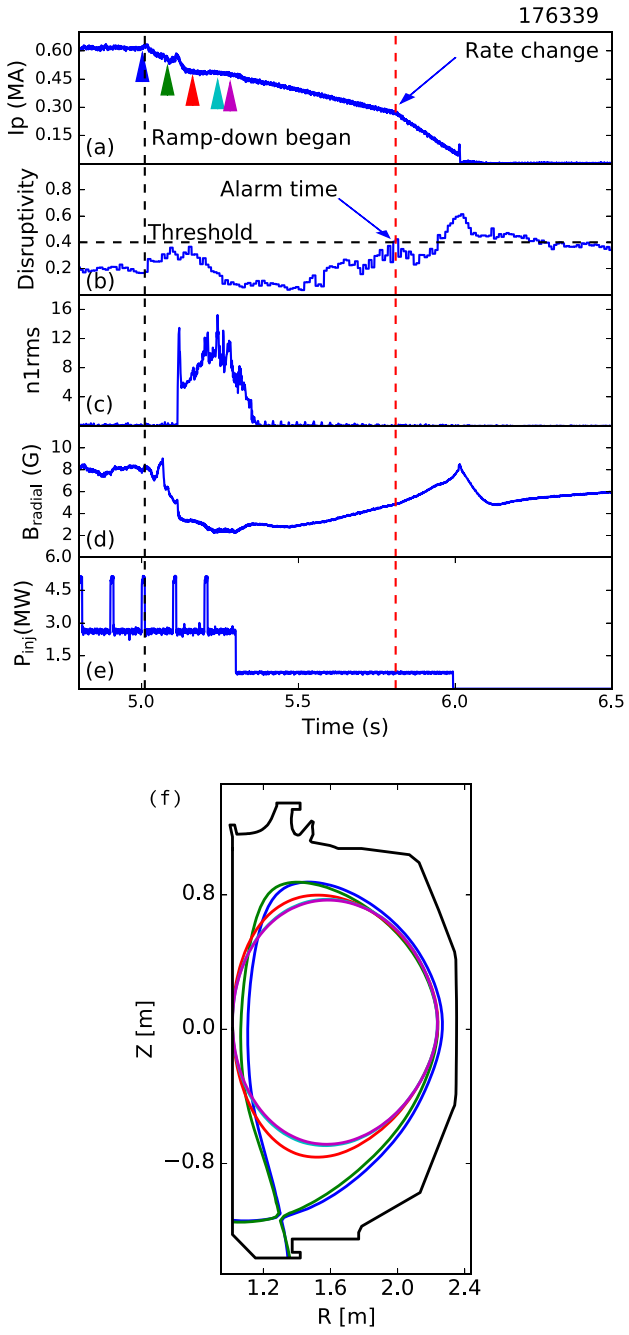


**FIG. 13.** Control ramp down speed based on machine learning prediction. The signals shown are plasma current(a), $n = 1$ RMS fluctuation amplitudes(c), radial magnetic field (d), and neutral beam power(e). The machine learning prediction, disruptivity, is shown in (b), and the horizontal dashed line is the threshold to change the ramp-down rate. The vertical black dashed line indicates the ramp down beginning. The vertical red dashed line is the time where disruptivity exceeds the threshold and triggers the ramp-down control. The different plasma shapes given by rtEFIT are shown in (f). The different colors indicate different times shown by triangular arrows in (a). These plots have 0.1 s increment start from $t = 5$ s, which clearly shows that the shape was changed from divertor shape into limiter shape.

threshold. After the shape change, the n = 1 instability disappeared. Meanwhile, the prediction of our algorithm decreased at first but grew again and finally exceeded the threshold at $t = 5.80$ s. This prediction triggered our control algorithm to ramp down faster. At the end of the ramp down, disruption still happened at $t = 6.01$ s, but at a very low plasma current ($I_p \approx 50$ kA) below the tolerance threshold $I_p \sim 0.3$ MA.

Finally, as a commentary of this experiment, the $B_{rad}$ calculation for shots between 176030 and 176912 was corrupted by a missing calibration for the F6A coil. This might cause the blindness of our algorithm to the $B_{rad}$ signal and lead to some error of disruptivity prediction in the real-time experiment.

## VI. TEARING MODE AVOIDANCE

The neutral beams (NB) are one of the primary sources of heating power for tokamak plasmas. Usually, the higher the NB power is, the greater the performance of the plasma will be. However, with higher NB power, the plasma will have a higher chance to cross the stability boundary and become unstable. Thus, it becomes challenging to keep the balance between having a high-performance plasma and keeping plasma stable. As an attempt to solve this problem, we developed a control algorithm based on the prediction of up-coming instabilities to optimize fusion performance by pushing the plasma to the limit of stability boundary, but not exceeding it.

### A. Tearing mode prediction

A tearing mode predictor was developed using the same method as the disruption predictor. During training, we used the tearing mode database explained in Sec. III A. The same strategy of splitting training and testing sets as disruption prediction was used. The training results are shown in Fig. 14. The median alarm time is not monotonically decreasing because when the threshold is too high, our algorithm will miss some tearing shots, and the median time would thus increase. If we set threshold = 0.16, algorithms were able to predict more than 90% of tearing modes with around 10% false positive. However, the median alarm time at this threshold is more than 600 ms, which is too long for the real-time experiment. When the threshold becomes higher, the median alarm time and FPR will decrease, but the TPR would decrease as well. When the threshold > 0.35, our algorithm would give less than 3% false positive prediction but would only have a 63% TPR with median alarm time around 380 ms. If we can keep the tearability around this level, we could potentially keep the plasma at the edge of the stability boundary and reach the maximum performance. Thus, during the real-time experiment, described in Sec. VI C, we expected to keep the tearability of plasma around 0.35. At this threshold, the distribution of alarm times is shown in Fig. 15. Here, the peak of distribution locates at $t_A = 210$ ms. As with disruption prediction, the four most important signals in tearing mode prediction are shown in Table IV.

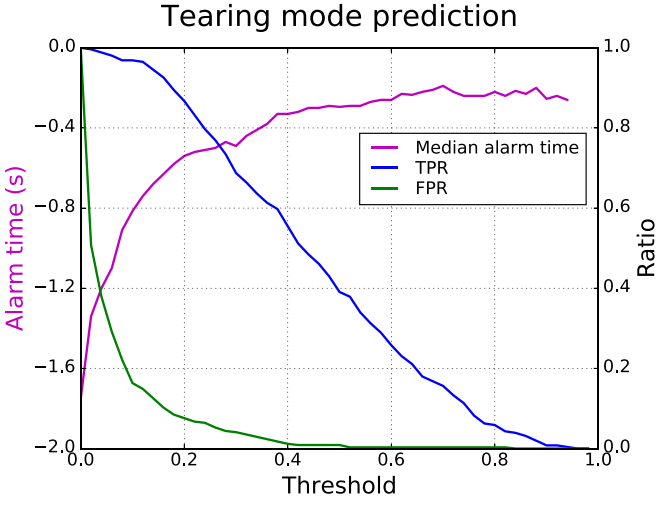


**FIG. 14.** TPR, FPR, and the median alarm times vs threshold for tearing mode prediction.

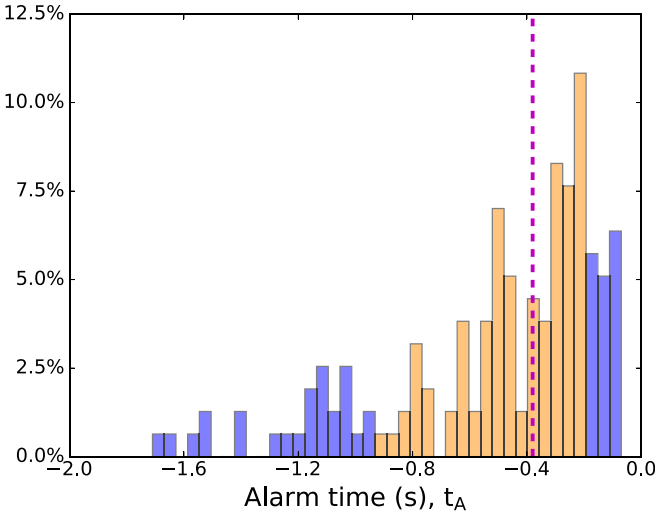


**FIG. 15.** The distribution of alarm times in tearing mode prediction at threshold = 0.35. The vertical dotted line represents the median alarm time 380 ms. Distributions within ±34% around the median value are shown by orange bins; the rest are shown by blue bins.

### B. Control algorithm and avoidance logic

Since there are too many different instabilities in tokamaks, it is impractical to predict all of them. Instead, we chose an important instability, tearing mode, to predict. Tearing modes are believed to be one of the most important causes and precursors of plasma disruption.[39] Based on the method in Sec. II and the tearing mode database described in Sec. III A, a machine learning algorithm was developed.

**TABLE IV.** Four most important signals in tearing mode prediction and their relative importance.

| Signal | Importance |
|---|---|
| $W_{MHD}$ | 0.276 |
| $\beta_t$ | 0.217 |
| $\langle n_e \rangle$ | 0.160 |
| n2rms | 0.158 |

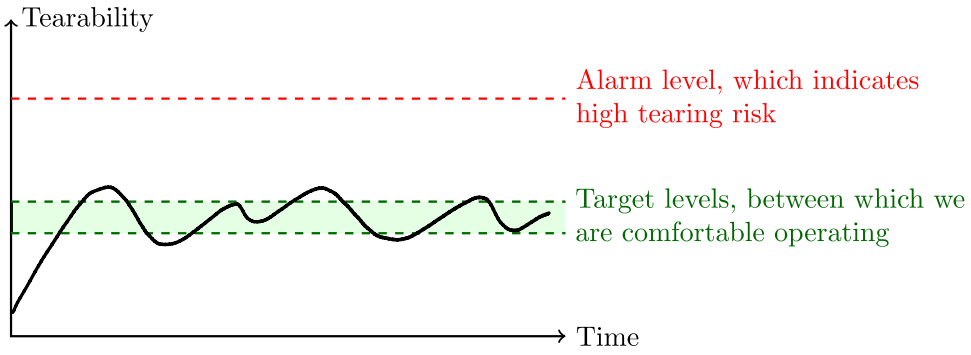


**FIG. 16.** Schematic diagram of two thresholds: the green lines are the target levels and the red line is the alarm level. The green region between the upper target level and lower target level is the target region. The black line indicates the expected tearability during the experiment.

The output of our machine learning algorithm, tearability, implies how likely tearing modes are to occur.

Using tearability as a metric of how unstable the plasma is, an instability avoidance logic was designed to command the neutral beam (NB) power by changing its duty cycle. First, we define two different tearability thresholds, as shown in Fig. 16. The higher threshold, shown by the red line, indicates an alarming level of the plasma. At this level, there is a very high chance that a tearing mode will occur shortly (for example, 250 ms). Thus, the alarm level could also be regarded as an approximated stability limit described in Sec. III A. When the tearability exceeds the alarm level, we believe the plasma is unstable at this stage, and some actions are needed to avoid tearing modes and a possible disruption. For example, electron cyclotron heating (ECH)[47–49] could be used to suppress the tearing mode by driving non-inductive current in the island and by heating the island.

The lower thresholds, composed of an upper target level and a lower target level indicated by the green lines, determine the target region where we expect the plasma to be operating. The target level could also be treated as a user-defined boundary within the approximated stability limit. Within this region, the likelihood of a tearing mode occurring is relatively low, and we believe that the plasma is stable. The higher target levels are, the risk of tearability exceeding the alarm level, due to fluctuation, would also be higher. Thus, the target levels indicate how much risk we are comfortable to operate at, as long as it is lower than the alarm level.

The logic of instability avoidance is the following: use NB power as the controlled parameter to keep tearability between the two target levels. During the flat-top stage of a shot, the machine learning algorithm keeps reading signals from real-time diagnostics and rtEFIT, predicting the tearability of the plasma. When the tearability is lower than the lower target level, the plasma is stable and far from tearing mode onset and disruption. However, in this regime, the plasma would have relatively low temperature and, therefore, low performance. Thus, the algorithm increases the NB power to enhance the performance of plasma. When the tearability is higher than the upper target level, the NB power is decreased to avoid instability. The workflow of this feedback control is shown in Fig. 17. The target levels, i.e., the user-defined boundaries, should be chosen such that fluctuation around it does not cause tearability to exceed the alarm level. Thus, by using this method, we could have the highest possible NB power, optimizing the plasma performance while keeping the plasma relatively stable.

### C. Real-time experiment

A real-time demonstrative experiment has been done at DIII-D. During these experiments, we set the target level around 0.35 and the alarm level at 0.4. At threshold = 0.35, the median time before tearing is 360 ms, which is relatively large. For the purpose of real-time control, we define the upper target level 0.35 and the lower target level 0.33, and we hope to keep the plasma tearability around these two levels. At threshold = 0.4, the median time before tearing would be 250 ms, and we believe that it is too close to tearing, and the plasma cannot be pushed further.

Two representative shots are shown in Fig. 18. First, in shot 176759, the tearing mode predictor was running in the background, but the control algorithm was turned off, and instead, the NB power is monotonically increasing. The purpose of this shot is to test how our machine learning algorithm works and how the plasma would respond. The signals and predictions are shown in Fig. 18(1). At the beginning of this shot, around $t = 2.4$ s, the NB power was low (≈2.5 MW), and tearability is below the target levels. The preprogrammed NB power, shown by the orange line in Fig. 18(1-b), was monotonically increased, though the actual NB power, shown by the blue line, was not the same as the coded power for technical reasons. The tearability, shown in Fig. 18(1-c), increased fast after the NB power increased, fluctuated around the alarm level. It first reached the alarm level around $t = 3.4$ s and greatly exceeded it after $t = 4.17$ s. In Figs. 18(1-d) and 18(1-e), we can see there is a relatively large n = 1 mode growing at $t = 3.9$ s and the mode locked at $t = 4.1$ s. Before mode locking, $\beta_N$ shown in Fig. 18(1-g) kept growing and reached maximum 2.47 at $t = 4.1$. After the lock mode occurs, $li$ began to grow, which means that the plasma crossed the stability limit and, finally, a disruption happened later at around $t = 5.1$s. Before mode locking, $\beta_N$ kept

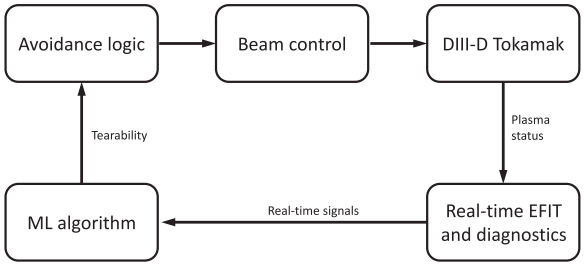


**FIG. 17.** Workflow of the feedback control of tearing modes.

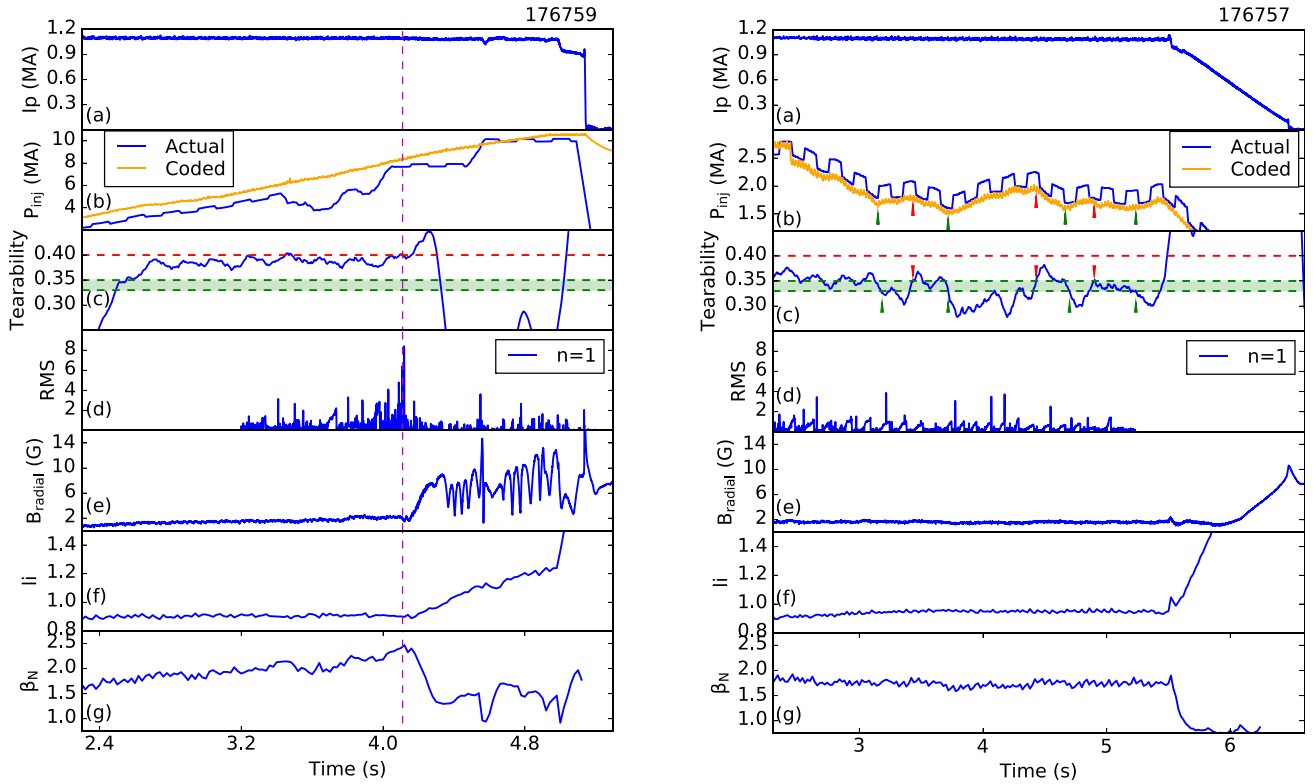


**FIG. 18.** Experimental results of tearing mode avoidance algorithm. Signals shown are (a) plasma current, (b) Neutral beam(NB) power, (d) n = 1 RMS amplitude fluctuation, (e) radial magnetic field, (f) normalize plasma inductance, and (g) normalized beta. The machine learning prediction, tearability, is shown in (c). The coded NB power is shown in orange and the actual NB power is shown in blue. The alarm level and target levels are shown by red and green dashed lines, respectively, in (c). The vertical magenta dashed line in 1 indicates the time of mode locking, represented by the sudden drop of n = 1 mode. (1) Comparative shot and (2) controlled shot.

increasing and reached the maximum 2.45 at $t = 4.1$ s. However, after mode locking, even though NB power kept increasing, $\beta_N$ dropped drastically. It worth mentioning that mode locking occurred before tearability exceeded the alarm level. It is because the tearability is only a probabilistic indicator of tearing mode, not deterministic. Thus when tearability was close to the alarm level, plasma was already likely to be unstable, and tearing mode could happen during that period. One more feature worth noticing is that the tearability grew at first and then dropped after the lock mode occurred. This is because our algorithm is predicting modes in the future; thus, tearability would drop after the mode actually occurred.

Next, in shot 176757, the control algorithm was turned on. At the beginning of the shot around $t = 2.4$ s, the NB power started at ≈2.5 MW but was controlled based on the tearability during the flat-top region. The details of this shot are shown in Fig. 18(2). In this particular shot, our control setting was when the tearability is larger than the upper target level (0.35), the NB power will be decreased; when the tearability is smaller than the lower level 0.33, the NB power will be increased. This was designed so that the tearability would fluctuate within the green shaded area shown in Fig. 18(2-c). The moments when tearability triggered the NB power to increase are pointed out by green arrows, and the moments that it triggered the decrease in NB power are pointed out by red arrows. These moments are shown in both Figs. 18(2-b) and 18(2-c), illustrating how the control worked during this shot. Note that in order to better demonstrate the control process, the tearability has been smoothed using the local average in the plot. Thus, the triggered moments in Figs. 18(2-b) and 18(2-c) do not exactly coincide with each other. After $t = 5.50$ s, the ramp-down began and the control algorithm was turned off. By applying this control method, we can see that the n = 1 RMS fluctuation and lock mode signal, shown in Figs. 18(2-d) and 18(2-e), was relatively low (RMS < 4, lock mode < 2) and the *li* was well controlled. During the whole shot, $\beta_N$ kept relatively constant at around 1.75 till ramp-down.

Comparing the two shots above, it is clear that our algorithm has the potential to prevent the growth of tearing instability by monitoring the tearability prediction. Although $\beta_N$ in the controlled shot was roughly 30% lower than the peak $\beta_N$ in the comparative shot, the algorithm prevented $\beta_N$ from dramatic drop due to instabilities, which could be necessary for a long-pulse steady-state operation. On the other hand, the reduction of $\beta_N$ could be explained by the lack of accuracy of our algorithm. Notice that from Fig. 14, the TPR at upper target levels is around 65%, which means even the tearability remains around this level, roughly 35% of them could still have tearing mode in the long run. Due to this uncertainty, when tearability exceeds this level, the NB power needs to be decreased. In other words, a relatively large margin between the user-defined boundary and the actual stability boundary has to be added to overcome the shortcoming of uncertainty.

Finally, similar to the ramp-down experiment, this tearing mode avoidance experiment was also affected by the miss calculation of the $B_{rad}$ signal and might cause some error in the tearability prediction.

## VII. SUMMARY AND DISCUSSION

Analysis of machine learning algorithms using data from DIII-D discharges showed that an accurate disruption and tearing mode prediction could be made with a prediction window of 250 ms prior to the event. After comparing multiple predictors, it was concluded that using the random forest, extremely randomized tree, or bagging algorithms all created predictors with the best performance on the data used. These decision tree forests have a high rate of detection (true positives $\sim$ 90%) with a low number of misidentified non-disruptive discharges (false positives $\sim$ 10%). These percentages indicate that the decision tree forest can predict most disruptions with a minimal effect on the non-disruptive discharges. This suggests that machine learning algorithms using decision tree forests are helpful tools to develop a disruption avoidance and mitigation system, which is suitable for the conditions and operating environment at DIII-D.

The preliminary development of machine-learning-based real-time disruption predictors on DIII-D has also been demonstrated in this paper. Real-time prediction results show that our algorithms are able to predict disruptions before they happen. Then based on that disruption predictor, a ramp-down control algorithm has been developed which can predict disruption during the ramp-down phase and change the ramp-down speed in order to either avoid disruption or keep disruption current within the tolerance threshold.

For a large-scale tokamak like ITER at full current operation, only 1–2 disruption events per lifetime are allowed.[50] It requires an extremely high true positive rate, which could be hard to achieve by current machine learning algorithms. In order to solve this problem, a potential solution could be predicting the precursors of disruptions, instead of the disruption itself, to avoid instabilities. Thus, a tearing mode predictor has been developed and used to control the neutral beam power. By monitoring the tearability prediction, our algorithm has the potential to optimize the plasma performance to the stability boundary and preventing tearing instability from occurring. During the control process, $\beta_N$ would decrease as the price for stability. This could be overcome by increasing the accuracy of the algorithm in the future or combining more predictors for different instabilities.

Though this paper provides a good proof of principle test showing the capability of MLA in real-time control; there remains a lot more work to develop these algorithms to a form usable in fusion reactors. During the training process, our algorithm made use of thousands of shots from DIII-D, and almost half of them were disruptive shots. This information would not be available on ITER at full current operation. There are three possible methods to solve this problem. First, based on the ITER research plan,[50] ITER will first operate with low plasma current ($Ip = 7.5$ MA) to verify its main features, and then, the operation will be extended to the full plasma current ($Ip = 15$ MA). Thus, we could train the algorithm with low current and use some proper extrapolation method to predict disruption at high current. Second, as part of the future work, we will test these control systems on different scenarios in DIII-D and develop methods to extrapolate the control to difference machines, especially large scale fusion reactors like ITER. Though some studies suggested that MLA might have issues extrapolating to different machines,[26] others show that simple normalization of each signal by its "global numerical scale" could be enough to get great cross-machine performance.[15] The third method is to build a predictor from scratch,[51] i.e., train the algorithm adaptive following the chronological order of ITER operation.

## ACKNOWLEDGMENTS

This work has been carried out within the framework of the EUROfusion Consortium and has received funding through FuseNet from the Euratom research and training program 2014-2018 under Grant Agreement No. 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility, under Award Nos. DC-AC02-09Ch11466, DE-FC02-04ER54698, and DE-SC0015878. DIII-D data shown in this paper can be obtained in digital format by following the links at https://fusion.gat.com/global/D3D_DMP.



## REFERENCES

[1]M. Lehnen, K. Aleynikova, P. B. Aleynikov, D. J. Campbell, P. Drewelow, N. W. Eidietis, Y. Gasparyan, R. S. Granetz, Y. Gribov, N. Hartmann *et al.*, "Disruptions in ITER and strategies for their control and mitigation," J. Nucl. Mater. **463**, 39–48 (2015).
[2]A. H. Boozer, "Theory of tokamak disruptions," Phys. Plasmas **19**(5), 058101 (2012).
[3]V. Igochine, *Active Control of Magneto-Hydrodynamic Instabilities in Hot Plasmas* (Springer, 2015), Chap. 7, pp. 227–257.
[4]A. Kallenbach, M. Bernert, T. Eich, J. C. Fuchs, L. Giannone, A. Herrmann, J. Schweinzer, W. Treutterer, and the ASDEX Upgrade Team, "Optimized tokamak power exhaust with double radiative feedback in ASDEX upgrade," Nucl. Fusion **52**(12), 122003 (2012).
[5]A. Krämer-Flecken, K. H. Finken, V. S. Udintsev, H. Larue, and TEXTOR Team, "Heterodyne ECE diagnostic in the mode detection and disruption avoidance at TEXTOR," Nucl. fusion **43**(11), 1437 (2003).
[6]J. W. Berkery, S. A. Sabbagh, R. E. Bell, S. P. Gerhardt, and B. P. LeBlanc, "A reduced resistive wall mode kinetic stability model for disruption forecasting," Phys. Plasmas **24**(5), 056103 (2017).
[7]S. P. Gerhardt, D. S. Darrow, R. E. Bell, B. P. LeBlanc, J. E. Menard, D. Mueller, A. L. Roquemore, S. A. Sabbagh, and H. Yuh, "Detection of disruptions in the high-$\beta$ spherical torus NSTX," Nucl. Fusion **53**(6), 063021 (2013).
[8]J. V. Hernandez, A. Vannucci, T. Tajima, Z. Lin, W. Horton, and S. C. McCool, "Neural network prediction of some classes of tokamak disruptions," Nucl. Fusion **36**(8), 1009 (1996).
[9]D. Wroblewski, G. L. Jahns, and J. A. Leuer, "Tokamak disruption alarm based on a neural network model of the high-beta limit," Nucl. Fusion **37**(6), 725 (1997).

[10]G. Pautasso, S. Egorov, C. Tichmann, J. C. Fuchs, A. Herrmann, M. Maraschek, F. Mast, V. Mertens, I. Perchermeier, C. G. Windsor *et al.*, "Prediction and mitigation of disruptions in ASDEX upgrade," J. Nucl. Mater. **290**, 1045–1051 (2001).
[11]C. G. Windsor, G. Pautasso, C. Tichmann, R. J. Buttery, T. C. Hender, and JET EFDA Contributors, "A cross-tokamak neural network disruption predictor for the JET and ASDEX upgrade tokamaks," Nucl. Fusion **45**(5), 337 (2005).
[12]B. Cannas, A. Fanni, P. Sonato, M. K. Zedda, and JET-EFDA Contributors, "A prediction tool for real-time application in the disruption protection system at JET," Nucl. Fusion **47**(11), 1559 (2007).
[13]R. Yoshino, "Neural-net disruption predictor in JT-60U," Nucl. Fusion **43**(12), 1771 (2003).
[14]A. Sengupta and P. Ranjan, "Forecasting disruptions in the ADITYA tokamak using neural networks," Nucl. Fusion **40**(12), 1993 (2000).
[15]J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, "Predicting disruptive instabilities in controlled fusion plasmas through deep learning," Nature **568**, 526 (2019).
[16]R. J. Buttery, T. C. Hender, D. F. Howell, R. L. Haye, S. Parris, O. Sauter, C. G. Windsor, and JET-EFDA Contributors, "On the form of NTM onset scalings," Nucl. Fusion **44**(5), 678 (2004).
[17]G. A. Rattá, J. Vega, A. Murari, G. Vagliasindi, M. F. Johnson, P. C. De Vries, and JET EFDA Contributors, "An advanced disruption predictor for JET tested in a simulated real-time environment," Nucl. Fusion **50**(2), 025005 (2010).
[18]Y. Zhang, G. Pautasso, O. Kardaun, G. Tardini, X. D. Zhang, and the ASDEX Upgrade Team, "Prediction of disruptions on ASDEX upgrade using discriminant analysis," Nucl. Fusion **51**(6), 063039 (2011).
[19]J. Vega, S. Dormido-Canto, J. M. López, A. Murari, J. M. Ramírez, R. Moreno, M. Ruiz, D. Alves, R. Felton *et al.*, JET-EFDA Contributors,. "Results of the JET real-time disruption predictor in the ITER-like wall campaigns," Fusion Eng. Des. **88**(6–8), 1228–1231 (2013).
[20]S. Esquembri, J. Vega, A. Murari, M. Ruiz, E. Barrera, S. Dormido-Canto, R. Felton, M. Tsalas, and D. Valcarcel, "Real-time implementation in JET of the SPAD disruption predictor using MARTe," IEEE Trans. Nucl. Sci. **65**(2), 836–842 (2018).
[21]L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees* (CRC Press, 1984).
[22]S. Rasoul Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE Trans. Syst. Man, Cybern. **21**(3), 660–674 (1991).
[23]A. Murari, J. Vega, G. A. Rattá, G. Vagliasindi, M. F. Johnson, S. H. Hong, and JET-EFDA Contributors, "Unbiased and non-supervised learning methods for disruption prediction at JET," Nucl. Fusion **49**(5), 055028 (2009).
[24]T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining, inference, and prediction," Biometrics **9**(2), 305 (2002).
[25]R. Polikar, "Ensemble based systems in decision making," IEEE Circuits Syst. Mag. **6**(3), 21–45 (2006).
[26]C. Rea, R. Granetz, K. J. Montes, R. A. Tinguely, N. W. Eidietis, J. M. Hanson, and B. S. Sammuli, "Disruption prediction investigations using machine learning tools on DIII-D and alcator C-Mod," Plasma Phys. Controlled Fusion **60**, 084004 (2018).
[27]R. J. La Haye, L. L. Lao, E. J. Strait, and T. S. Taylor, "High beta tokamak operation in DIII-D limited at low density/collisionality by resistive tearing modes," Nucl. Fusion **37**(3), 397 (1997).
[28]O. Meneghini, S. P. Smith, L. L. Lao, O. Izacard, Q. Ren, J. M. Park, J. Candy, Z. Wang, C. J. Luna, V. A. Izzo *et al.*, "Integrated modeling applications for tokamak experiments with OMFIT," Nucl. Fusion **55**(8), 083008 (2015).
[29]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res. **12**, 2825–2830 (2011).
[30]J. R. Ferron, B. Penaflor, M. L. Walker, J. Moller, and D. Butner, "A flexible software architecture for tokamak discharge control systems," in *Proceedings of 16th International Symposium on Fusion Engineering* (IEEE, 1995), Vol. 2, pp. 870–873.
[31]J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning* (Springer Series in Statistics, New York, USA, 2001).
[32]G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in Neural Information Processing Systems* (2013), pp. 431–439.
[33]D. D. Boos, "Introduction to the bootstrap world," Stat. Sci. **18**(2), 168–174 (2003).
[34]L. Breiman, "Bagging predictors," Mach. Learn. **24**(2), 123–140 (1996).
[35]L. Breiman, "Random forests," Mach. Learn. **45**(1), 5–32 (2001).
[36]P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," Mach. Learn. **63**(1), 3–42 (2006).
[37]Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," J. Jpn. Soc. Artif. Intell. **14**(771–780), 1612 (1999).
[38]T. Fawcett, "An introduction to ROC analysis," Pattern Recognit. Lett. **27**(8), 861–874 (2006).
[39]P. C. De Vries, M. F. Johnson, B. Alper, P. Buratti, T. C. Hender, H. R. Koslowski, V. Riccardo, and JET-EFDA Contributors. "Survey of disruption causes at JET," Nucl. Fusion **51**(5), 053018 (2011).
[40]J. A. Stillerman, T. W. Fredian, K. A. Klare, and G. Manduchi, "MDSplus data acquisition system," Rev. Sci. Instrum. **68**(1), 939–942 (1997).
[41]J. R. Ferron, M. L. Walker, L. L. Lao, H. S. John, D. A. Humphreys, and J. A. Leuer, "Real time equilibrium reconstruction for tokamak discharge control," Nucl. Fusion **38**(7), 1055 (1998).
[42]T. C. Luce, M. R. Wade, P. A. Politzer, S. L. Allen, M. E. Austin, D. R. Baker, B. Bray, D. P. Brennan, K. H. Burrell, T. A. Casper *et al.*, "Long pulse high performance discharges in the DIII-D tokamak," Nucl. Fusion **41**(11), 1585 (2001).
[43]B. Cannas, A. Fanni, E. Marongiu, and P. Sonato, "Disruption forecasting at JET using neural networks," Nucl. Fusion **44**(1), 68 (2004).
[44]P. A. Politzer, G. L. Jackson, D. A. Humphreys, T. C. Luce, A. W. Hyatt, and J. A. Leuer, "Experimental simulation of ITER rampdown in DIII-D," Nucl. Fusion **50**(3), 035011 (2010).
[45]P. C. de Vries, T. C. Luce, Y. S. Bae, S. Gerhardt, X. Gong, Y. Gribov, D. Humphreys, A. Kavin, R. R. Khayrutdinov, C. Kessel *et al.*, "Multi-machine analysis of termination scenarios with comparison to simulations of controlled shutdown of ITER discharges," Nucl. Fusion **58**(2), 026019 (2018).
[46]N. W. Eidietis, W. Choi, S. H. Hahn, D. A. Humphreys, B. S. Sammuli, and M. L. Walker, "Implementing a finite-state off-normal and fault response system for disruption avoidance in tokamaks," Nucl. Fusion **58**(5), 056023 (2018).
[47]K. Hoshino, M. Mori, T. Yamamoto, H. Tamai, T. Shoji, Y. Miura, H. Aikawa, S. Kasai, T. Kawakami, H. Kawashima *et al.*, "Avoidance of q a= 3 disruption by electron cyclotron heating in the JFT-2M tokamak," Phys. Rev. Lett. **69**(15), 2208 (1992).
[48]I. G. J. Classen, E. Westerhof, C. W. Domier, A. J. H. Donné, R. J. E. Jaspers, N. C. Luhmann, Jr.,, H. K. Park, M. J. van de Pol, G. W. Spakman, M. W. Jakubowski *et al.*, "Effect of heating on the suppression of tearing modes in tokamaks," Phys. Rev. Lett. **98**(3), 035001 (2007).
[49]B. Esposito, G. Granucci, P. Smeulders, S. Nowak, J. R. Martin-Solis, L. Gabellieri, and ECRH teams, "Disruption avoidance in the Frascati tokamak upgrade by means of magnetohydrodynamic mode stabilization using electron-cyclotron-resonance heating," Phys. Rev. Lett. **100**(4), 045006 (2008).
[50]P. C. De Vries, G. Pautasso, D. Humphreys, M. Lehnen, S. Maruyama, J. A. Snipes, A. Vergara, and L. Zabeo, "Requirements for triggering the iter disruption mitigation system," Fusion Sci. Technol. **69**(2), 471–484 (2016).
[51]J. Vega, A. Murari, S. Dormido-Canto, R. Moreno, A. Pereira, A. Acero, and JET-EFDA Contributors, "Adaptive high learning rate probabilistic disruption predictors from scratch for the next generation of tokamaks," Nucl. Fusion **54**(12), 123001 (2014).